ボトムアップ手法を用いた系統樹構築の近似アルゴリズム(計算機科学の理論とその応用)

AUTHOR(S):

前村, 一哉; 小野, 廣隆; 定兼, 邦彦; 山下, 雅史

CITATION:

前村, 一哉 ...[et al]. ボトムアップ手法を用いた系統樹構築の近似アルゴリズム(計算機科学の理論とその応用). 数理解析研究所講究録 2007, 1554: 238-242

ISSUE DATE:

2007-05

URL:

http://hdl.handle.net/2433/80947

数理解析研究所講究録
第1554巻 2007年 238-242

ボトムアップ手法を用いた系統樹構築の近似アルゴリズム

前村 一哉 (Kazuya Maemura) *　　小野 廣隆 (Hirotaka Ono) †
定兼 邦彦 (Kunihiko Sadakane) †　　山下 雅史 (Masafumi Yamashita) †
* 九州大学大学院 システム情報科学府
Dept. of Electrical Engineering and Computer Science, Kushu University
† 九州大学大学院 システム情報科学研究院
Dept. of Electrical Engineering and Computer Science, Kushu University

概説

系統樹は異なる生物種が進化の過程において，どのように分岐したかを木として表したものである．系統樹においてラベル付けされた葉は生物種に，根はすべての種の共通祖先に，内部節点は葉となる種の共通祖先に対応している．

本稿では，葉となる種の集合Sと3つの種からなる3点系統樹の集合Tを入力として与えられたとき，Tに含まれるできるだけ多くの3点系統樹の祖先・子孫関係を満たす木を構築する問題を扱う．この問題はNP完全問題であることが証明されており，近似アルゴリズムの立場から研究されている [2,3]．本稿では、既存の近似アルゴリズム [3] を改良したアルゴリズムを提案し，提案したアルゴリズムと元のアルゴリズムに関する定理の証明を行う．

1　はじめに

現存するすべての生物種はある共通の祖先から進化しており，それらの間には何らかの進化的な関係があると考えられている．系統樹とは，進化の過程において共通祖先からどのように分岐してきたかという生物種間の進化的関係を木構造で表現したものである．系統樹は根付き非順序木であり，その木において一意にラベル付けされた葉は (現存する) 生物種に，根はすべての生物種の共通祖先に，内部節点は葉となる生物種の祖先となる種に対応している．図1は系統樹の例である (http://www.christiananswers.net/home.html より)．

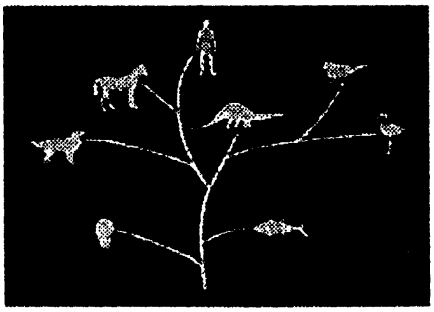

図 1: 系統樹の例

数多く存在する既知の種に対する系統樹を直接考えるのは現実的ではない．そのため，一般的には数種類 (3,4種類) の種からなる部分的な系統樹を用いて全体の木の構築を考える．著者の研究では，3種類の生物種からなる3点系統樹と呼ばれるものから木を構築する問題を取り扱う．

2　準備

本節では準備として．問題の入力となる3点系統樹と3点系統樹を入力とした系統樹構築の問題について述べる．

2.1　3点系統樹

i,j,kを種とすると，3点系統樹は図2のような木で表すことができる．この木は$\{i,j\} < \{i,k\}$や

$\{i,j\} < \{j,k\}$ のように表記される．このとき $\{i,j\}$ は種 i と j の LCA(共通祖先で最も根から遠い祖先，*Lowest Common Ancestor*) にあたるノードを表しており，同様に $\{i,k\}$ は種 i と k の LCA にあたるノードを表している．$\{i,j\} < \{i,k\}$ は i と j の LCA は i と k の LCA の正統な子孫であることを示している．また，$\{i,j\} < \{i,k\}$ や $\{i,j\} < \{j,k\}$ は $(\{i,j\},k)$ とも表記されることがある．本稿では，これ以降の3点系統樹の表記は $(\{i,j\},k)$ を用いることにする．

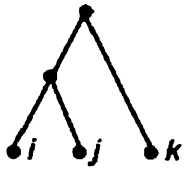

図 2: 3点系統樹の例

2.2 系統樹構築に関する問題

系統樹を構築する問題の入力として，一般的には S と T が与えられる．S は生物種の集合を，T は S の3つの種からなる3点系統樹の集合を表している (以降は $|S| = n, |T| = m$ とする)．"3点系統樹における祖先・子孫関係" は木を構築するための "制約" と考えることができる．

具体的に図2のような3点系統樹が T に含まれる1つの要素として与えられているときを考える．この場合，i と j は同じ部分木の葉にならなくてはならない．これは，もし i と j が異なる部分木の葉ならば，i と j の LCA は根ということになり，図のような祖先・子孫関係を満たさなくなるためである．

著者の研究で扱う問題は，S と T が与えられたときに T に含まれる制約を満たす木を構築するというものである．T のすべての制約を満たす木を構築できるかという決定問題は，多項式時間で解けることが既に示されている [1]．

また，T の制約をできる限り多く満たすような木を構築する問題も研究されており，これは NP 完全問題であることが証明されている [2, 3]．この問題は MICT(*the Maximum Inferred Consensus Tree problem*) や MCTT(*the Maximum Consensus Tree from rooted Triples problem*) と呼ばれ，近似アルゴリズムの立場から研究されている．この問題では，いかに上手く制約を満たす木を構築するかが重要となる．

3 既存の研究

本節では既存の研究の論文 [3] において提案されている近似アルゴリズムについて述べる．次節では，本アルゴリズムを改良したアルゴリズムについて説明する．

3.1 アルゴリズム Best Pair Merge First

アルゴリズム Best Pair Merge First(以降 PM と書く) の流れを以下に示す．初期集合として，それぞれ n 種類の異なる種に一意にラベル付けされた n 個の節点 (木) の集合 τ がある．集合 τ は1つの木のみを含む集合になるまで，2つの木を選んで結合する．

定義 3.1 $V(t_i)$ は木 t_i のもつ葉 (種) の集合を表す．

このアルゴリズムでは，結合する2つの木を選ぶ基準として $e\_score(V(t_i), V(t_j))$ という関数を用いる．ここで $t_i, t_j \in \tau$ とする．各々の反復で最大の $e\_score(V(t_i), V(t_j))$ となるような2つの木を選び，結合する．アルゴリズム PM は e_score が最大になるような2つの木を選んで結合することによって，満たす3点系統樹の数ができるだけ多い木をヒューリスティックに構築している．アルゴリズムの詳細を表1に示す．

定義 3.2

- $w(V(t_i), V(t_j))$: $(\{i,j\},x)$ となる3点系統樹の数．ただし，$i \in V(t_i)$, $j \in V(t_j)$, $x \in S \backslash \{V(t_i) \cup V(t_j)\}$ とする．

• **Algorithm Best Pair Merge First**
(1)$\tau = \{t_l \mid 1 \leq l \leq n\}$．$t_l$ は葉 l だけを含む木
(2)**while** $\|\tau\| > 1$ **do**
(2-1) τ から $e\_score(V(t_i), V(t_j))$ が最大になるような t_i と t_j を選択
(2-2) 共通祖先となる内部節点を加えて t_i と t_j を結合して新たな木を構築
endwhile
(3) 構築された木を返す

表 1: アルゴリズム Best Pair Merge First

- $p(V(t_i),V(t_j))$: t_i と t_j を選んだ時に $(\{i,j\},x)$ に反する3点系統樹の数．より具体的には $(\{i,x\},j)$, $(\{j,x\},i)$ となる3点系統樹の数．ただし，$i \in V(t_i)$, $j \in V(t_j)$, $x \in S\backslash\{V(t_i) \cup V(t_j)\}$ とする．
- $t(V(t_i),V(t_j))$: $(\{i,j\},x)$ となる3点系統樹の数．ただし $i \in V(t_i), j \in V(t_j), x \in S\backslash\{i,j\}$ とする．

e_score として，以上の3つの関数を組み合わせた6つの関数を定義する．分子に $w(V(t_i),V(t_j))$, $w(V(t_i),V(t_j)) - p(V(t_i),V(t_j))$ を用いたものをそれぞれ if-penalty が False, True であると呼ぶ．また，分母に 1, $w(V(t_i),V(t_j)) + p(V(t_i),V(t_j))$, $t(V(t_i),V(t_j))$ を用いたものをそれぞれ Ratio-type が 0, 1, 2 であると呼ぶ．関数 *e_score* をまとめたものが表2である．この表では $w(V(t_i),V(t_j))$ を w, $p(V(t_i),V(t_j))$ を p, $t(V(t_i),V(t_j))$ を t と表記している．

	Ratio-type		
if-penalty	0	1	2
False	w	$\frac{w}{w+p}$	$\frac{w}{t}$
True	$w-p$	$\frac{w-p}{w+p}$	$\frac{w-p}{t}$

表 2: 関数 $e\_score(V(t_i),V(t_j))$ の組合せ表

論文においては具体的な近似比と時間計算量は述べられていない．この論文の著者が行った計算機実験では，実験のインスタンスの7割以上で，論文 [2] で提案されている2つの近似アルゴリズムよりも良い結果となっている．また，アルゴリズム PM において関数 *e_score* を計算する回数は $O(n^3)$ 回であるので，時間計算量はおよそ $O(mn^3)$ であると言える．

4 提案手法

本節では，前節で紹介したアルゴリズム PM を改良したアルゴリズムについて述べる．また，著者が行ったアルゴリズム PM と本節のアルゴリズムを比較するための実験およびその結果，さらに定理とその証明について述べる．

4.1 アルゴリズム Best Pair Merge with Reconstruction

前節で述べたアルゴリズム PM は木の集合から関数 *e_score* を最大にする2つの木を選び，結合して新しい木を構築するものであった．そのため，初めに2つの木 t_x と t_y が結合して次に t_z が結合するときに，この順序で結合した木が最も多くの3点系統樹を満たすとは限らない．つまり，アルゴリズム PM において図3(1) のような木になるときに図3(2) や (3) のような木に構築した方が満たす3点系統樹が多い場合があると言える．本稿で提案するアルゴリズムは PM において，この点を考慮したアルゴリズムである．

図 3: t_x, t_y, t_z が結合する木の例

定義 4.1 関数 $Sat(t_i)$ を木 t_i が満たす入力 T に含まれる3点系統樹の数とする．

アルゴリズム Best Pair Merge with Reconstruction(以降 PMR を書く) の流れを以下に示す．初期集合として，それぞれ n 種類の異なる種に一意にラベル付けされた n 個の節点 (木) の集合 τ がある．τ が1つの木のみを含むまで2つの木を選んで結合する点と木を選ぶ基準として関数 *e_score* を用いる点は PM と同じである．

アルゴリズムの任意の反復で *e_score* を最大にする木が t_i と t_j であるときを例に考える．t_i は2つの木 t_{i1} と t_{i2} が結合してできた木とする．同様に t_j は2つの木 t_{j1} と t_{j2} が結合してできた木とする．このとき，図4の各々の木の Sat の値を計算する．この5つの中で最大の Sat の値となる木の形に木を再構築する (PM では，すべて t_α の形に構築されることになる)．アルゴリズムの詳細を表3に示す．

4.2 実験結果の比較

任意の S と T を入力として，アルゴリズム PM で構築される木を T_{PM}，PMR で構築される木を T_{PMR}，最適解の木を T_{opt} とする．

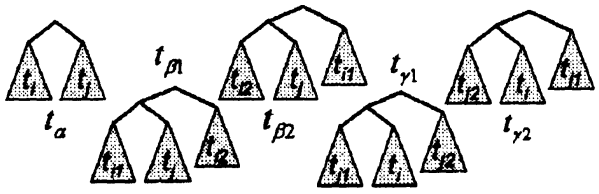

図 4: $t_\alpha, t_{\beta 1}, t_{\beta 2}, t_{\gamma 1}, t_{\gamma 2}$

Average	PM00	PM10	PM01	PM11	PTM02	PM12
n=15,m=50	1.199	1.106	1.087	1.103	1.254	1.089
n=15,m=100	1.222	1.106	1.117	1.113	1.320	1.120
n=15,m=200	1.190	1.084	1.064	1.082	1.254	1.067
n=15,m=300	1.198	1.091	1.076	1.077	1.237	1.082

Average	PMR00	PMR10	PMR01	PMR11	PMR02	PMR12
n=15,m=50	1.190	1.106	1.087	1.103	1.225	1.089
n=15,m=100	1.207	1.100	1.111	1.105	1.250	1.111
n=15,m=200	1.187	1.083	1.077	1.075	1.186	1.079
n=15,m=300	1.194	1.079	1.073	1.076	1.196	1.080

表 4: 実験結果の比較 (平均値)

Worst	PM00	PM10	PM01	PM11	PTM02	PM12
n=15,m=50	1.357	1.226	1.226	1.226	1.500	1.188
n=15,m=100	1.327	1.173	1.182	1.182	1.452	1.182
n=15,m=200	1.306	1.158	1.119	1.119	1.429	1.130
n=15,m=300	1.293	1.145	1.114	1.131	1.345	1.167

Worst	PMR00	PMR10	PMR01	PMR11	PMR02	PMR12
n=15,m=50	1.310	1.226	1.226	1.226	1.500	1.188
n=15,m=100	1.300	1.173	1.182	1.182	1.370	1.182
n=15,m=200	1.306	1.137	1.119	1.119	1.286	1.130
n=15,m=300	1.293	1.145	1.114	1.131	1.288	1.167

表 5: 実験結果の比較 (最悪値)

- **Algorithm Best Pair Merge with Reconstruction**

(1)$\tau = \{t_l | 1 \leq l \leq n\}$. t_l は葉 l だけを含む木
(2)**while** $|\tau| > 1$ **do**
(2-1) τ から $e\_score(V(t_i), V(t_j))$ が最大になるような t_i と t_j を選択
(2-2) $Sat(t_\alpha), Sat(t_{\beta 1}), Sat(t_{\beta 2}), Sat(t_{\gamma 1}), Sat(t_{\gamma 2})$ の値を計算する
(2-3) Sat の値が最大になる木に構築する
endwhile
(3) 構築された木を返す

表 3: アルゴリズム Best Pair Merge with Reconstruction

アルゴリズム PM と PMR の構築する木の満たす 3 点系統樹の数を比較するために，実装してその実験結果を比較した．$n = 15$ で固定し，m =50, 100, 200, 300 で行った．0 から 14 までの整数を種としてランダムに発生させ，その 3 つずつを組として 3 点系統樹を作成した．各々の m で乱数の種を変えて 10 回ずつ行った．

表 4 と表 5 は実験結果をまとめたものである．PM01 は，アルゴリズム PM で e_score は if-penalty が False で Ratio-type が 1 のときの結果であることを表している．表 4 の数値は各々の結果の $\frac{Sat(T_{opt})}{Sat(T_{PM})}$，または $\frac{Sat(T_{opt})}{Sat(T_{PMR})}$ の平均値を示している．表 5 の数値は各々の結果の $\frac{Sat(T_{opt})}{Sat(T_{PM})}$，または $\frac{Sat(T_{opt})}{Sat(T_{PMR})}$ の最悪の値を示している．

4.3 定理とその証明

定理 4.1 アルゴリズム PM と PMR における木の結合は，集合 τ に含まれる木に対する関数 e_score の値のみで決まる．また，e_score は木 t_i が持つ葉 (種) から計算される．任意の入力 S, T において，$Sat(T_{PM}) \leq Sat(T_{PMR})$ が成り立つ．

証明 アルゴリズム PMR の任意の反復において $e\_score(V(t_i), V(t_j))$ が最大となるとき，以下の 2 通りに場合分けができる．

(i) 再構築が行われない場合

この場合，図 4 の t_α が構築される．これは PM と同じなので，$Sat(T_{PM}) = Sat(T_{PMR})$ である．

(ii) 再構築が行われる場合

$V(t_\alpha) = V(t_{\beta 1}) = V(t_{\beta 2}) = V(t_{\gamma 1}) = V(t_{\gamma 2})$ は明らか．これより木の再構築後の反復では，e_score の引数は再構築が行われても変わることはない．これより，T_{PM} と T_{PMR} は t_α が再構築されている部分のみが異なっている．

$Sat(T_{PM})$ と $Sat(T_{PMR})$ の大小関係は再構築された部分木の満たす 3 点系統樹によって決まる．再構築が行われる場合には，その部分木の満たす 3 点系統樹の数は $Sat(t_\alpha)$ よりも多いので，$Sat(T_{PM}) < Sat(T_{PMR})$ となる．

(i),(ii) より，$Sat(T_{PM}) \leq Sat(T_{PMR})$ が成り立つことが示された． □

5 まとめと今後の課題

本稿では系統樹と3点系統樹，系統樹構築の問題について簡単に述べた．そして，既存の研究のアルゴリズムであるアルゴリズムBest Pair Merge Firstを元にしたアルゴリズムBest Pair Merge wiht Reconstructionを提案し，実験結果の比較と定理の証明を行った．

今後の課題としては，本稿で提案したアルゴリズムの改良が挙げられる．また，新たなアルゴリズムを考案してその考察，解析などを行うことも挙げられる．

参考文献

[1] A.V.Aho, Y.Sagiv, T.G.Szymanski and J.D.Ullman, "Inferring a tree from lowest common ancestors with an application to the optimizationof relational expressions", *SIAM Journal of Computing*, Vol.10, No.3, pp.405-421, 1981.

[2] L.Gasieniec, J.Jansson, A.Lingas and A.Östlin, "On the Complexity of Constructing Evolutionary Trees", *Journal of Combinatorial Optimization*, Vol.3, pp.183-197, 1999.

[3] B.Y.Wu, "Constructing the Maximum Consensus Tree from Rooted Triples", *Journal of Combinatorial Optimization*, Vol.8, No.1, pp.29-39, 2004.